1. Complete Calculations

$$(x(t), y(t)) = (\frac{X(t)}{Z(t)}, \frac{Y(t)}{Z(t)})f$$

Using 3d rotation matrix (Image from google search)

$$\text{Z-axis: } \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$\text{X-axis: } \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \bullet \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

$$\text{Y-axis: } \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \bullet \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

Roll

In the case of rotation with angular velocity Ω_Z about the camera Z axis.

Let (x,y) be the image position on the depth Z_0.

In 2D, As we know from the rotation matrices, if a vector is rotated anti-clockwise around the origin by θ degrees, its new coordinates will be

$$(cos(\theta x) - sin(\theta y), sin(\theta x) + cos(\theta y))$$

Then if the vector is rotated clockwise around the origin with angular velocity Ω_Z by t seconds, then the new image coordinates will be $(\cos(\Omega_z x) - \sin(\Omega_z y), \sin(\Omega_z x) + \cos(\Omega_z y))$.

Therefore, If the camera coordinates of a 3D scene point at time t = 0 are (X_0, Y_0, Z_0), then at time t with camera rolling at clockwise direction, the scene point will be at:

$$(X(t), Y(t), Z(t)) = (cos(\Omega_z t X_0) - sin(\Omega_z t Y_0),\ sin(\Omega_z t X_0) + cos(\Omega_z t Y_0),\ Z_0)$$

Vector on Z direction will not change with t, the coordinates of projected image is not depending on the Z_0. Therefore we can ignore the Z(t) in this case. We apply

$$(x(t), y(t)) = (X(t), Y(t)) * f$$

The image coordinate of the point is a function of t, namely,

$$(x(t), y(t)) = (cos(\Omega_z t x) - sin(\Omega_z t y), sin(\Omega_z t x) + cos(\Omega_z t y)) f$$

Taking the derivative with respect to t at t = 0 yields an image velocity vector (v_x,v_y) :

$$\frac{d}{dt} x(t)|_{t=0} = (-sin(\Omega_z t x)\Omega_z x - cos(\Omega_z t y)\Omega_z y)|_{t=0} = -\Omega_z y$$
$$\frac{d}{dt} y(t)|_{t=0} = (cos(\Omega_z t x)\Omega_z x - sin(\Omega_z t y)\Omega_z y)|_{t=0} = \Omega_z x$$
$$(v_x, v_y) = \frac{d}{dt}(x(t), y(t))|_{t=0}$$
$$= \Omega_z(-y, x)$$

Tilt

Rotation with angular velocity Ω_x about the camera X axis.

Let (x,y) be the image position on the depth Z_0.

As we know from the rotation matrices, if the camera coordinates of a 3D scene point at time t = 0 are (X_0, Y_0, Z_0), then at time t with camera pitching at clockwise direction, the scene point will be at

$$(X(t), Y(t), Z(t)) = (X_0,\ cos(\Omega t)Y_0 - sin(\Omega t)Z_0,\ sin(\Omega t)Y_0 + cos(\Omega t)Z_0)$$
$$\frac{d}{dt} x(t)|_{t=0} = \frac{d}{dt}(\frac{X(t)}{Z(t)}) f|_{t=0} = -\frac{X_0\Omega(Y_0 cos(0) - Z_0 sin(0))}{(Y_0 sin(0) + Z_0 cos(0))^2} f = -\frac{\Omega X_0 Y_0}{Z_0^2}$$
$$\frac{d}{dt} y(t)|_{t=0} = \frac{d}{dt}(\frac{Y(t)}{Z(t)}) f|_{t=0} = -\frac{\Omega(Y_0^2 + Z_0^2)}{(-Y_0 sin(0) + Z_0 cos(0))^2} f = -\frac{\Omega(X_0^2 + Z_0^2)}{Z_0^2} f$$

Since X_0/Z_0=x/f,

$$(v_x, v_y) = (-\Omega_x \frac{xy}{f}, -\Omega_x(1 + (\frac{x}{f})^2) f)$$

Pan

Rotation with angular velocity Ω_x about the camera Y axis.

Let (x,y) be the image position on the depth Z_0.

As we know from the rotation matrices, if the camera coordinates of a 3D scene point at time t = 0 are (X_0, Y_0, Z_0), then at time t with camera pitching at clockwise direction, the scene point will be at

$$(X(t), Y(t), Z(t)) = (\ cos(\Omega t)X_0 + sin(\Omega t)Z_0,\ Y_0,\ \ - sin(\Omega t)X_0 + cos(\Omega t)Z_0)$$

$$\frac{d}{dt}x(t)|_{t=0} = \frac{d}{dt}(\frac{X(t)}{Z(t)})f|_{t=0} = -\frac{\Omega(X_0^2 + Z_0^2)}{(-X_0 sin(0) + Z_0 cos(0))^2}f = \frac{\Omega(X_0^2 + Z_0^2)}{Z_0^2}f$$

$$\frac{d}{dt}y(t)|_{t=0} = \frac{d}{dt}(\frac{Y(t)}{Z(t)})f|_{t=0} = -\frac{Y_0\Omega(-X_0 cos(0) - Z_0 sin(0))}{(-X_0 sin(0) + Z_0 cos(0))^2}f = \frac{\Omega X_0 Y_0}{Z_0^2}$$

Similar to the calculation of Tilt, the motion fild can be derived to

$$(v_x, v_y) = (\Omega_y(1 + (\frac{x}{f})^2)f,\ \Omega_y\frac{xy}{f})$$

2. Compute 3D Points in World Coordinates

The code is in Compute_point_cloud.m

The details of the implementation are commented in the code.

After loading the image, I iterate every pixel in the image one by one, compute its (X,Y) by solve

$$Ax = B$$

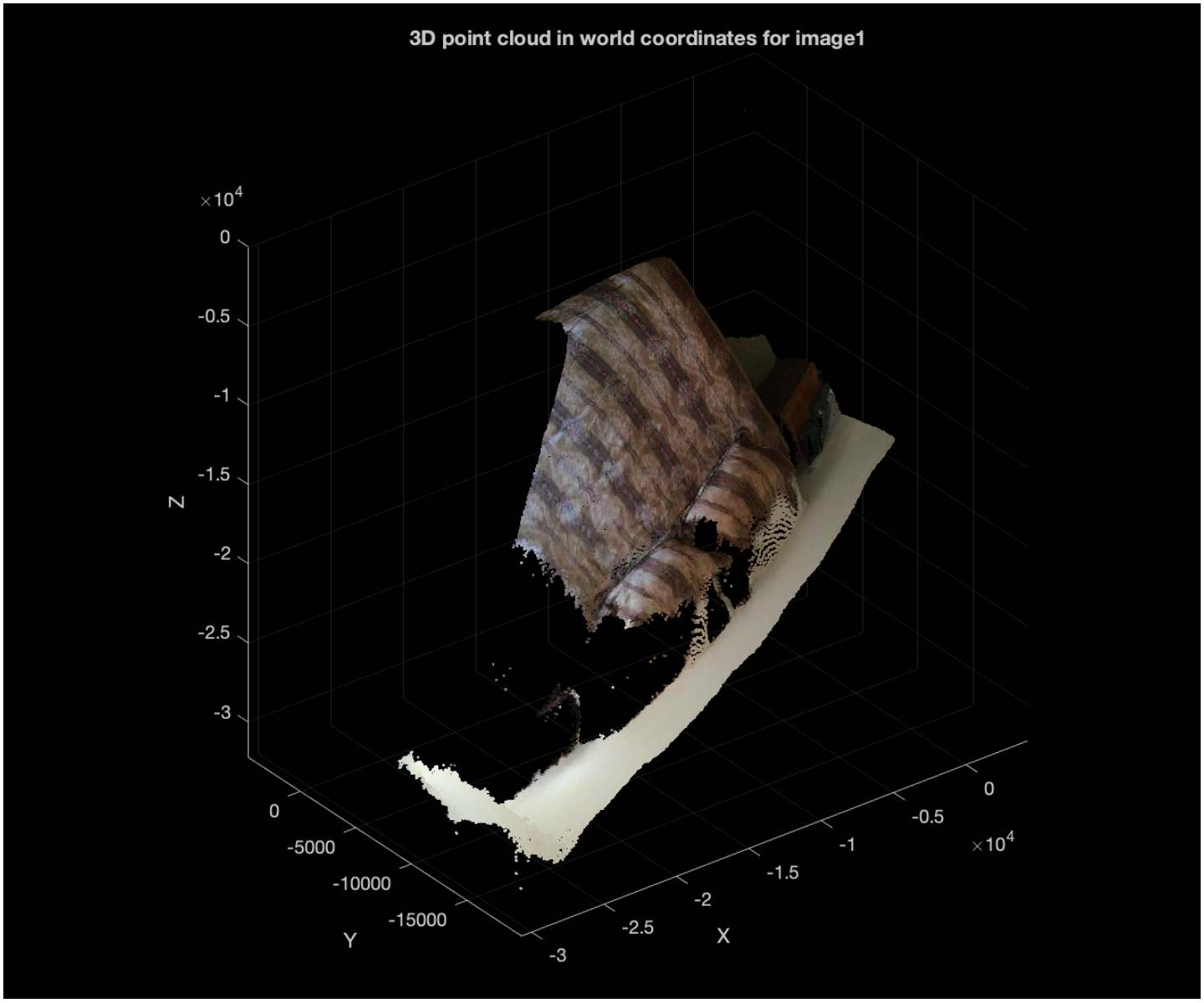
where B=[xw,xy,w]' and A= Intrinsics_Matrix and Extrinsic_Matrix.

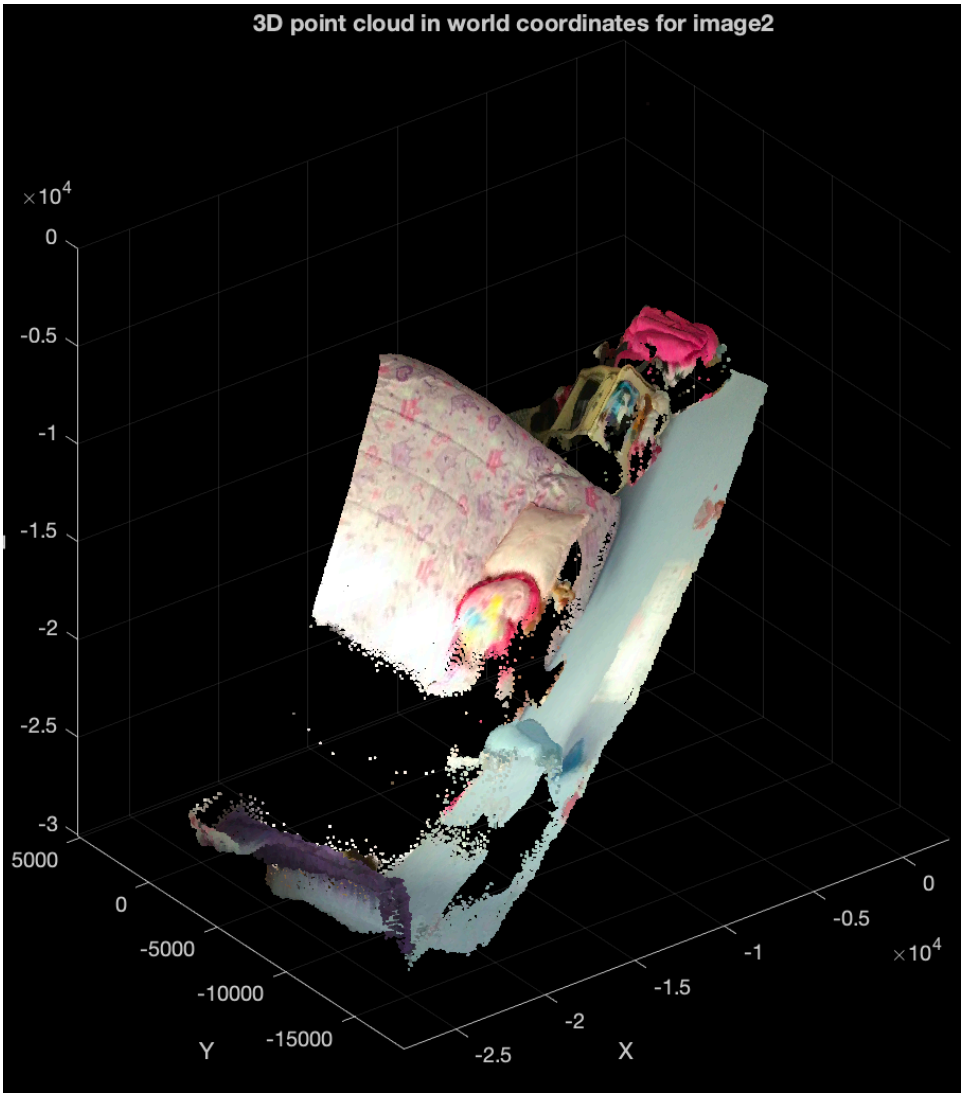
Since we know Z, let B=[x,y,1'] and 3*1 matrix x we got using linesolve need to be normalized by

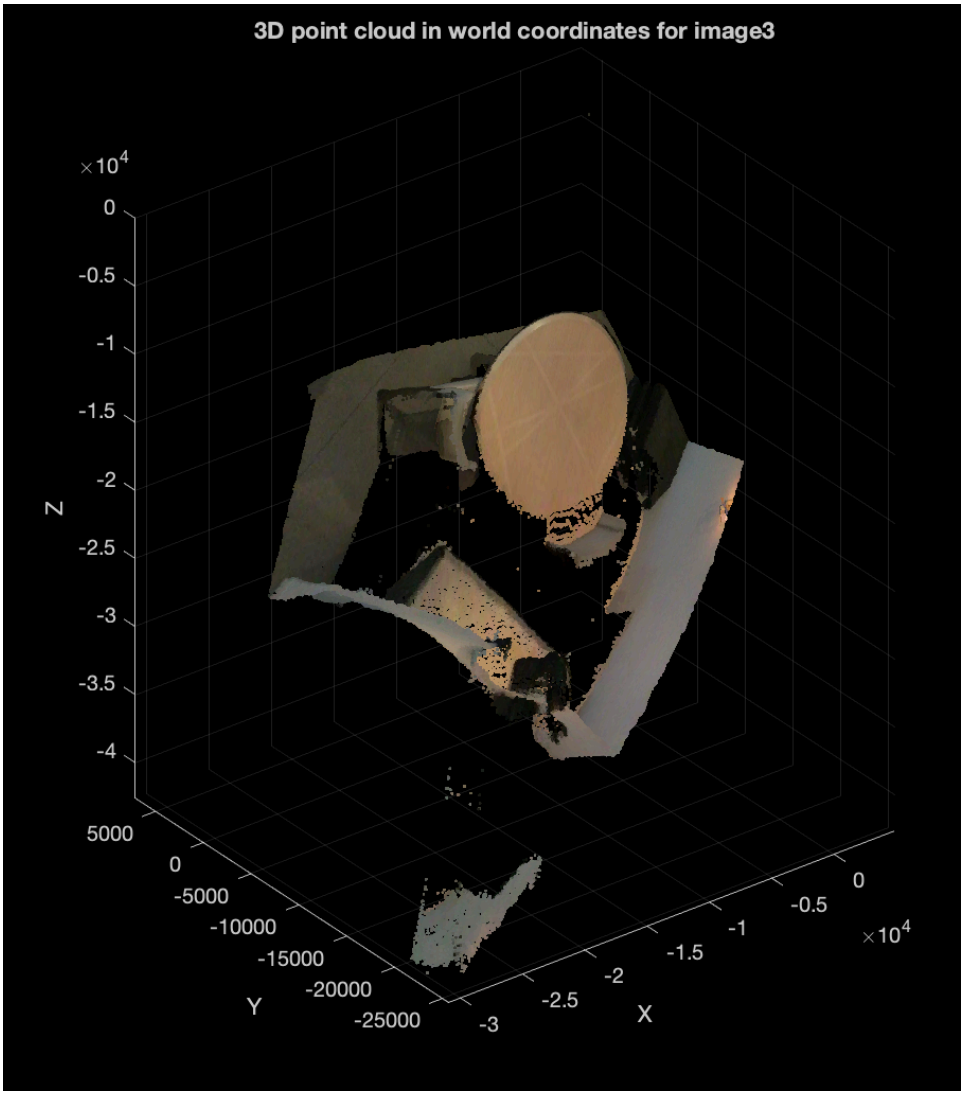
```
x=x/x(3)*z
```

Then x will be 3d coordinates that correspoind to (x,y) in graph.

The results for image 1,2 and 3


3D point cloud in world coordinates for image1
×10^4
0
-0.5
-1
-1.5
-2
-2.5
-3
Z
0
-5000
-10000
-15000
Y
-3
-2.5
-2
-1.5
-1
-0.5
0
×10^4
X


3D point cloud in world coordinates for image2
×10^4
0
-0.5
-1
-1.5
-2
-2.5
-3
5000
0
-5000
-10000
-15000
Y
-2.5
-2
-1.5
-1
-0.5
0
×10^4
X



For part 2 and part 3, I will not switch x,y position in plot because it may cause confussion and make codes complicated.

The point coordinates (x,y) i stores in the matrix are mathematically mean the y-th row, x-th column. However, Matlab reads (x,y) as x-th row, y-th column, the x,y axis of plot that generated later are all switched. Sorry for the confussion I made.

3. New Coordinates in Camera Reference Frame

Code for this part is function **compute_new_3dpoints** in compute_2d_projection.m and also the function **compute_point_cloud_camera** in compute_point_cloud_camera.m

The details of the implementation are commented in the code.

The main idea is to first map the 2d pixels points to 3D scene points in camera reference frame, then apply rotation and translation.

Intrinsics_Matrix contains all information to map 3d coordinates to 2d pixel points,

K=

fmx	s	px
0	fmy	py
0	0	1

The coordinate of 3D points in camera frame can be computed by

$$X = (x - px) * Z / fm_x;$$
$$Y = (y - py) * Z / fm_y;$$
$$Z$$

where (x,y) is corresponding point in pixel coordinates

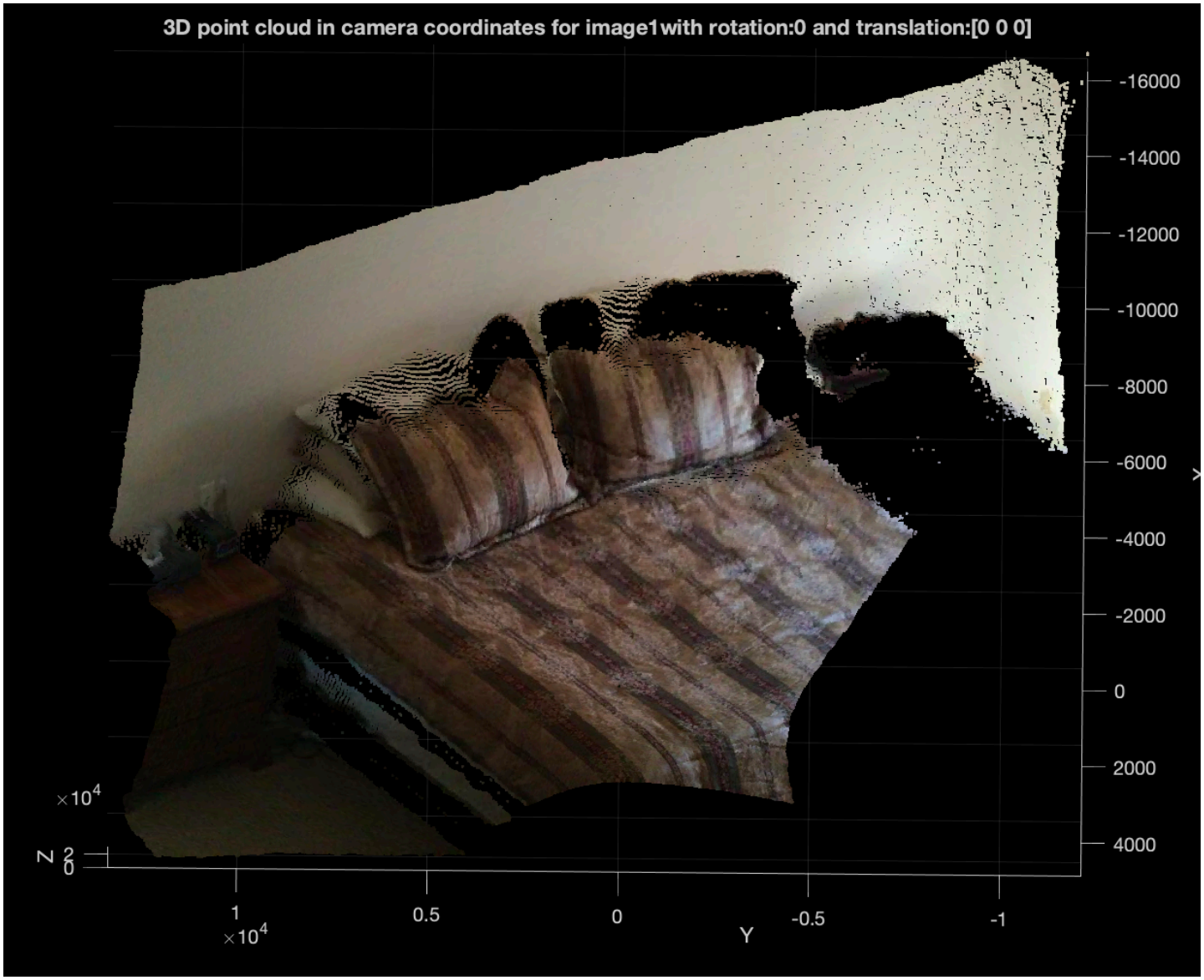
Then do the rotation according to the camera by multiply (X,Y,Z)' and rotation matrix.

For translation, we can add the translation vector directly to (X,Y,Z)

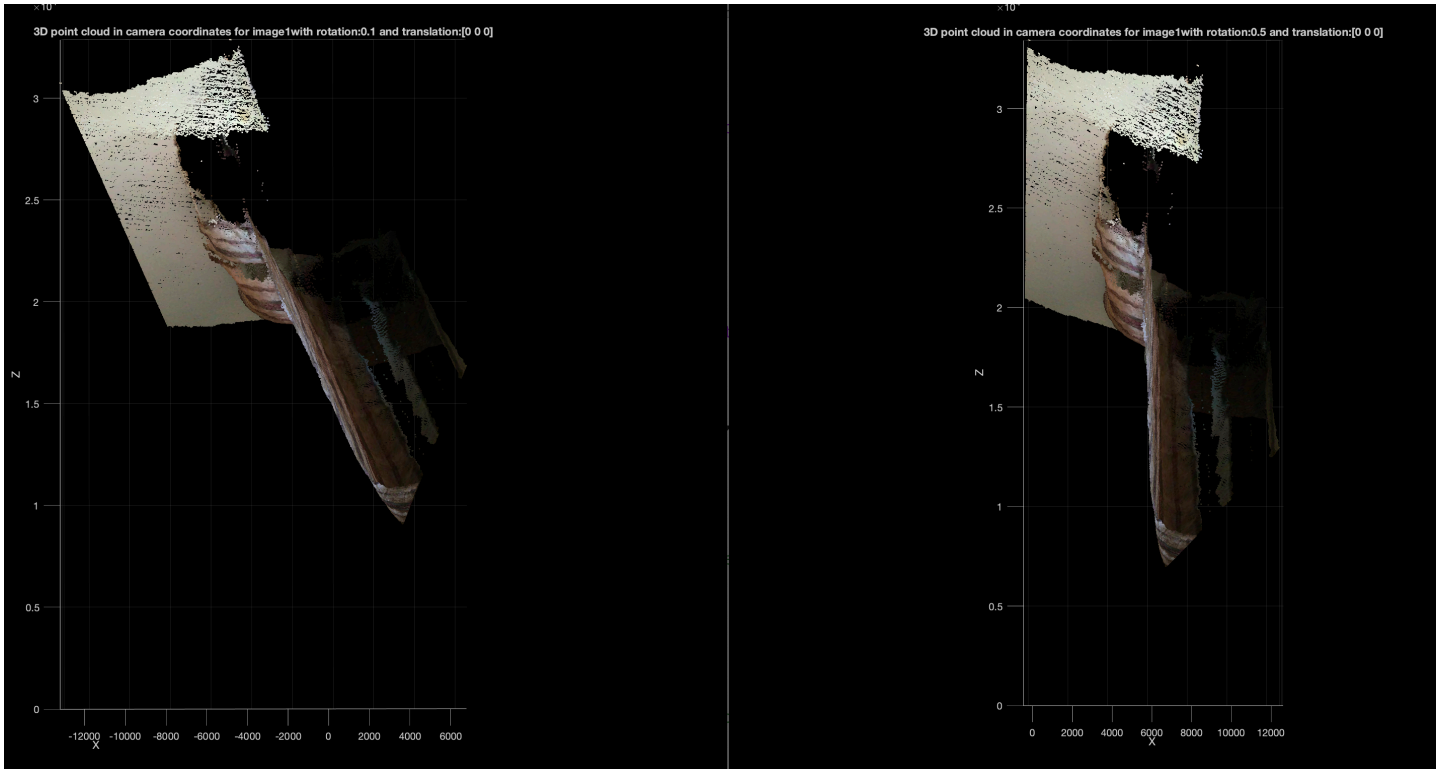
Here are some example result: that base on image 1:

Example: The results with no rotation and no translation:

As we can see the 3d point cloud is align to the camera coordinates in XY-plane



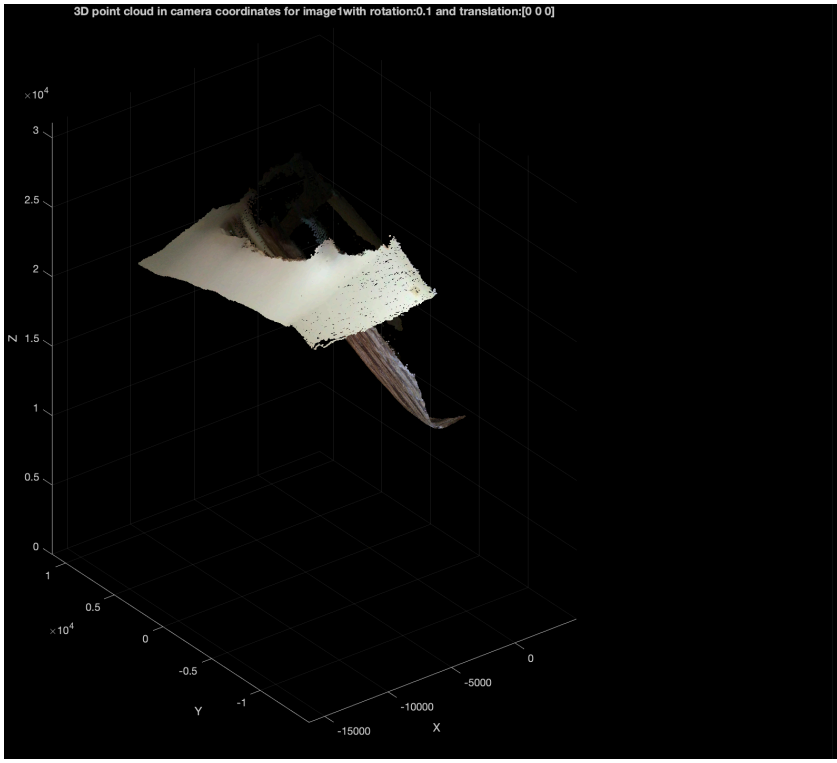
Example: Rotation around Y axis 0.1 and 0.5 result: XY-plane comparison

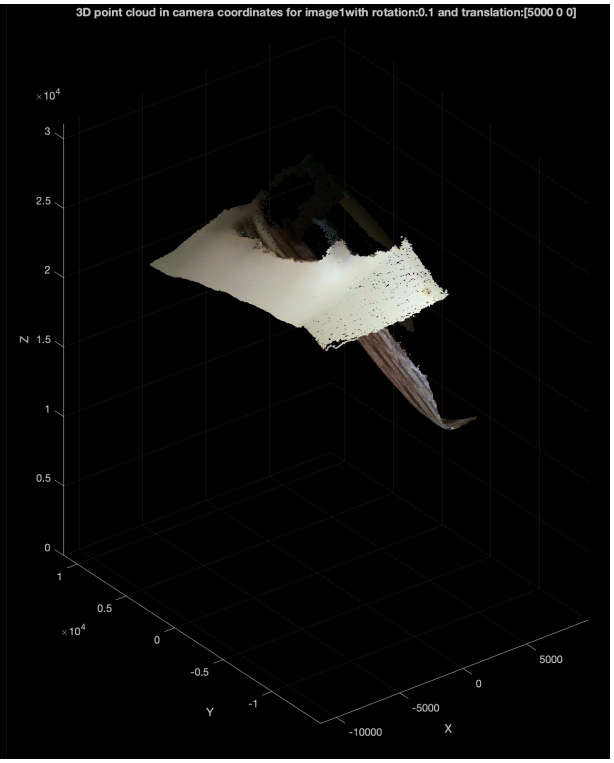


Example: Rotation around X axis 0.1 and 0.5 result: YZ-plane comparison

Example: translation [0 0 0] and [5000 0 0] result:

As we can see from x axis, the graph move toward direction of (1,0,0) 5000 units.


3D point cloud in camera coordinates for image1with rotation:0.1 and translation:[0 0 0]
×10^4
3
2.5
2
Z 1.5
1
0.5
0
1
0.5
×10^4
0
-0.5
-1
Y
0
-5000
-10000
-15000
X


3D point cloud in camera coordinates for image1with rotation:0.1 and translation:[5000 0 0]
×10^4
3
2.5
2
Z 1.5
1
0.5
0
1
0.5
×10^4
0
-0.5
-1
Y
5000
0
-5000
-10000
X

4. Sequence of Rotations and Translations

The point coordinates (x,y) i stores in the matrix are mathematically mean the y-th row, x-th column. However, Matlab reads (x,y) as x-th row, y-th column, the x,y axis of plot that generated later are all switched. Sorry for the confussion I made.

In part 3, we have computed the 3D coordinates after rotation and translation. Now we only need to multiply Intrinsics_Matrix and (X,Y,Z) to obtain (wx,wy,w), then compute

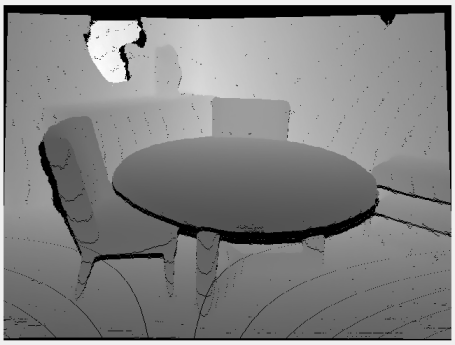
(x,y,1)=(wx,wy,w)/w to obtain the pixel points.

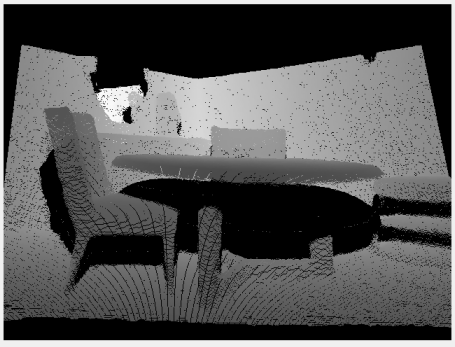
After rotation and translation, some points may fell out of the image dimension . I filtered out the points that is out of frame, and display the image with remaining points.

Example result:

RGB Image and Depth Image Rotation around Y axis and translation along the X axis in step 1 and step 6:

The camera is heading up and moving down

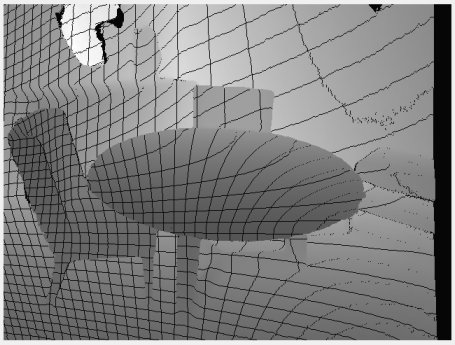
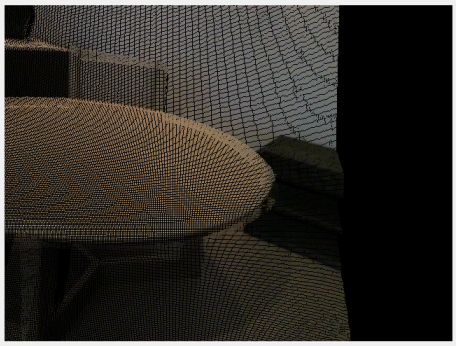
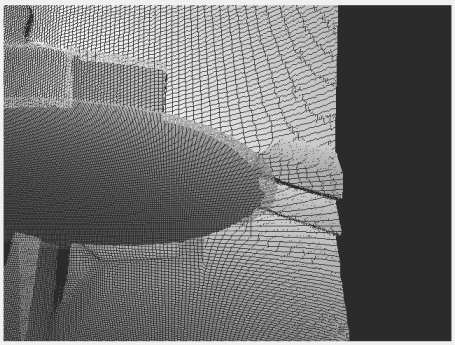





RGB Image and Depth Image Rotation around X axis and translation along the Z axis in step 1 and step

RGB Image and Depth Image Rotation around X axis and translation along the Z axis in step 1 and step 6:

The camera is heading right and moving closer to the object

RGB Image and Depth Image Rotation around Z axis and translation along the Y axis in step 1 and step 6:

The camera is rotating and moving right.